Catopen

Be careful of function's implicit use of environment variables

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-19

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 9468 bytes

Attack Categories	• Path spoofing or confusion problem • Malicious Input
Vulnerability Categories	• Format string • Buffer Overflow • Indeterminate File/Path • TOCTOU - Time of Check, Time of Use • Unconditional
Software Context	• National Language Support • File Path Management
Location	
Description	The catopen() function is vulnerable to manipulations that will substitute a different catalog file for the expected one. The catopen(char *name, int oflag) function is used to open a message catalog and returns a catalog descriptor. The first argument is the name of the message catalog to be opened. If it contains a "/", then the name is a path name, otherwise it is a base name. The second input is used to specify locale differences. The function implicitly uses the values of environment variables, even when the name argument contains a "/". It can do vaguely printf()-like substitutions on the filename. It does things like replacing %L with the value of the LANG environment variable. See also Catgets rule. There is also an internal buffer weakness by which a buffer overflow could occur through the setting of an environment variable. A privileged application which uses catopen() could be made to execute arbitrary code by an unprivileged local user.
APIs	**FunctionName** | **Comments**

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

	<table><tr><td>catopen</td><td>opens catalog; verify location</td></tr></table>
Method of Attack	These functions open a locale-based message catalog that is used by other functions such as catgets() and gettext(). It is possible for a user to install a malformed message catalog, then manipulate the NLSPATH environment variable to feed the bad catalog to the users program. The improper use of routines like catgets() and gettext() then expose the program to potential arbitrary code execution. The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results.
Exception Criteria	
Solutions	<table><tr><th>Solution Applicability</th><th>Solution Description</th><th>Solution Efficacy</th></tr><tr><td>Particularly applicable to setuid root programs where invoker can control execution environment.</td><td>Validate that catopen() will return an authentic message catalog. Place the message database (i.e., set of directories containing message catalog files) in a secure, trusted directory. Make certain that an attacker can not manipulate the program environment (not necessarily an option for a setuid program) and that the information</td><td>Effective, but may not be easy to implement properly.</td></tr></table>

		used to locate the particular message catalog file is validated before catopen() is called. Specifying a fully qualified catalog path containing "/" would work, but largely defeats the purpose of using a message catalog. The alternative is to examine NLSPATH and related environment variables to confirm that they correspond only to the expected secure directories. This also requires that the message catalog locations be constrained, with those constraints known to the program at compilation time.	
	Generally applicable.	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and	Does not resolve the underlying vulnerability but limits the false sense of security given by the check.

	identity cannot be assured, but it does help to limit the false sense of security given by the check.	
Generally applicable.	Limit the interleaving of operations on catalogs from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
Generally applicable.	Limit the spread of time (cycles) between the check and use of a resource.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
Generally applicable.	Recheck the resource after the use call to verify that the action was taken appropriately.	Effective in some cases.

SignatureDetails	Presence of the catopen function.

Examples of Incorrect Code

```
catopen("mymessages.txt");
printf("%s", catgets(...));

/* This example is bad because
someone can set the following in
their environment:

NLSPATH="/usr/local/lib/locale/%L/
%N"
LANG="../../../../etc/
master.passwd\0" #### (where \0
represents the null character)
```

This would get through even if one validated NLSPATH, but would probably cause some amount of the master.passwd file to be printed, if the process had rights to open and read that file. */

Examples of Corrected Code

```
#include <string>
```

```
/*A simple function that checks
for parent directory references in
the path */
/* A check for \0 might also be
worthwhile */
bool IsNLSPathSafe()
{
std::string nlspath =
getenv("NLSPATH");
return (nlspath.find("..") < 0);
}

// Verify an expected secure path
will be searched - check NLSPATH,
LANG, etc.
if (!IsNLSPathSafe())
exit(EXIT_FAILURE);
nl_catd catd =
catopen("MyCatalog", 0);

// Ensure safe usage of retrieved
text
char *text = catgets(catd, 2, 10,
"Default text.");
printf("%s", text);
strncpy(buffer, text,
bufferSize);
```

Source Reference	• http://www.linuxsecurity.com/content/view/102556/103/
Recommended Resource	"Wide Spread UNIX Vulnerability"[3] by Dave Wreski posted to Bugtraq - Original report on catalog vulnerability (poorly formatted)
Discriminant Set	**Operating System** — ; **Language** —

Cigital, Inc. Copyright

1. mailto:copyright@cigital.com